System Analysis Methods

Objectives

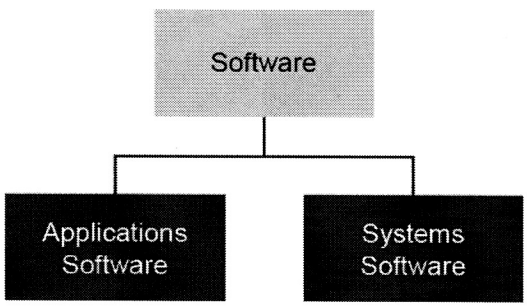
- Describe the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development
- Describe the relative merits and drawbacks of different methodologies and when they might be used

Software development

- What is software?
- Give some examples of:
  - software that you have written
  - software that you use
  - software that you know about

Types of software

- Software refers to all programs that run on a computer
  - It falls into one of two categories:

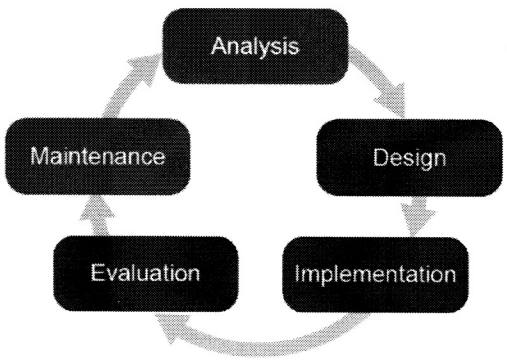


Software development

- Software development, from the initial idea to the final product, follows distinct stages
  - Analysis
  - Design
  - Implementation (programming, testing and installation)
  - Evaluation
  - Maintenance

The systems development life cycle

- This is often shown as a circle:



Analysis

- In this stage, a **systems analyst** gathers information about
  - What the current system does, if there is one
  - What the new system needs to do
- To do this, the systems analyst may:
  - **Interview** people who will use the software
  - Use **questionnaires** to get information from large groups of people
  - **Observe** how the current system works
  - Look at existing **documentation**

Output from Analysis

- The systems analyst will produce a document called something like

  "System Specification" or "User Requirements"
- This defines what the system will do, but not how it will do it
- The specification is a vital document!
- It is used to create the design, and to evaluate the finished product

Design

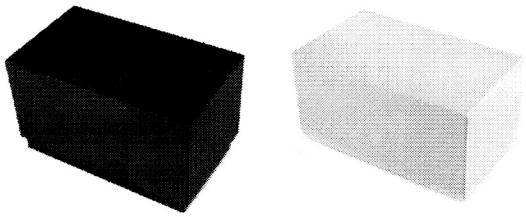
- The software design will include:
  - A description of the data: data type, format, and validations
  - Database design if appropriate
  - Input screens
  - Output screens and reports
  - How the data will be processed
  - How the software will be tested

Implementation

- This stage includes:
  - Coding and testing the software
  - Writing user and technical documentation
  - Installing the software for the user

Testing strategies – black box

- Black box testing is carried out independently of the code used in the program
- It looks at the program specification and creates a set of test data that covers all the inputs, outputs and program functions



Testing strategies – white box

- White box testing depends on the code logic
- Tests are devised which test each path through the code at least once
  - Why might you need both of these types of testing?

Alpha testing

- This is carried out by the software developer's in-house team and by the user
- It can reveal errors or omissions in the definition of the system requirements
- The user may discover that the system does not do exactly what they wanted
  - Why might this happen? it is not in development enough to consider all of these aspects

Beta testing

- This is used when commercial software is being developed (e.g. MS Windows, MS Word, Sage Accounts, etc.)
- The software is given to a number of potential users, who agree to use the software and report any faults
  - How is this helpful?

Evaluation

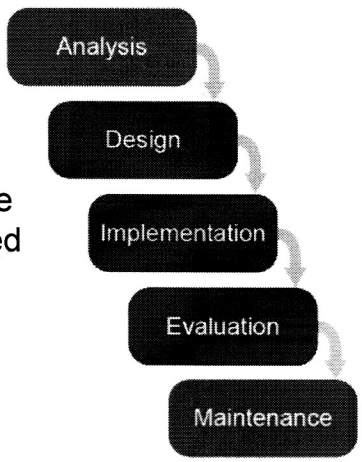
- Does it all work?
- The user now needs to test every aspect of the software to make sure it does what it is supposed to do
- It will be evaluated against the original specification document
- This stage is also called **Acceptance testing**

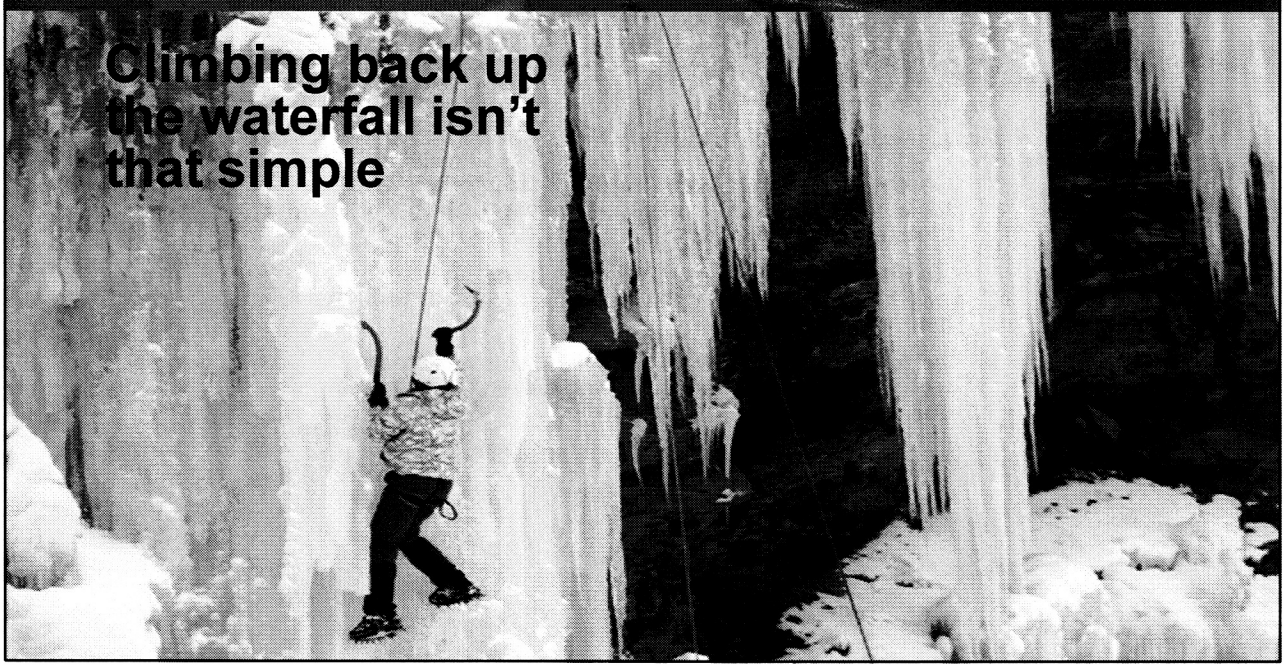
Maintenance

- Three types of maintenance:
  - **Corrective maintenance**. Bugs will usually be found when the software is put into action, no matter how thoroughly it was tested
  - **Adaptive maintenance**. Over time, user requirements will change and the software will have to be adapted to meet new needs
  - **Perfective maintenance**. Even if the software works well, there may be ways of making it even better – faster, easier to use, more functionality
- And now the cycle begins all over again!

The waterfall model

- As in the lifecycle model, each stage is completed and documented before the next is begun
- The customer does not see the end product until it is completed
- Any change to be made often means the project has to be started again





Advantages of this model

- The model is simple to understand and use
- Each stage is separate and self-contained with well defined outcomes and written documentation
- This makes the project relatively straightforward to manage
- The model works well for smaller projects where requirements are very well understood

Disadvantages of the model

- There is not much user involvement after the Analysis stage, when the Specification document is agreed
- No working software is produced until late in the cycle
- The user is presented with the finished product and if it is not quite what was required, it is generally too late to make changes

When to use the waterfall model

- This model is suitable when:
  - The requirements are very clear and fixed
  - There are no ambiguous requirements
  - The technology is well understood
  - The project is short

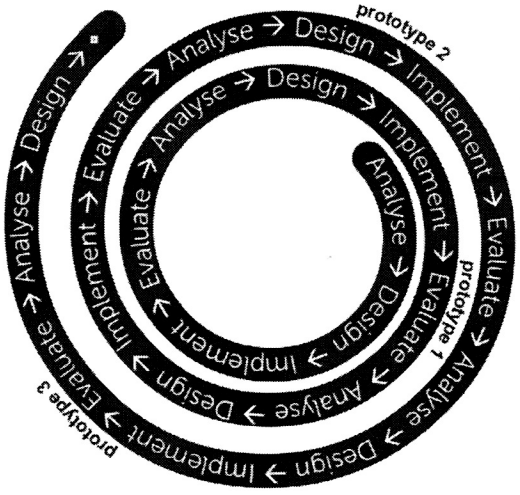
Demonstration

Worksheet 1

- Now try **Task 1**, **Questions 1 and 2** on the worksheet

Activation

The spiral model



Spiral model

- The four basic steps of analysis, design, implementation (i.e. programming and testing) and evaluation are followed
- The software project passes through these phases repeatedly
- Each successive loop round the spiral generates a new, more refined prototype until the software meets all the requirements

Advantages of the spiral model

- The well-defined steps make the project easy to manage
- Software is produced at an early stage so problems and issues can be identified early
- The user gives feedback on each prototype and any required changes can be made early in the process
- Added functionality can be added during the process
- The end result is more likely to be what the user wants

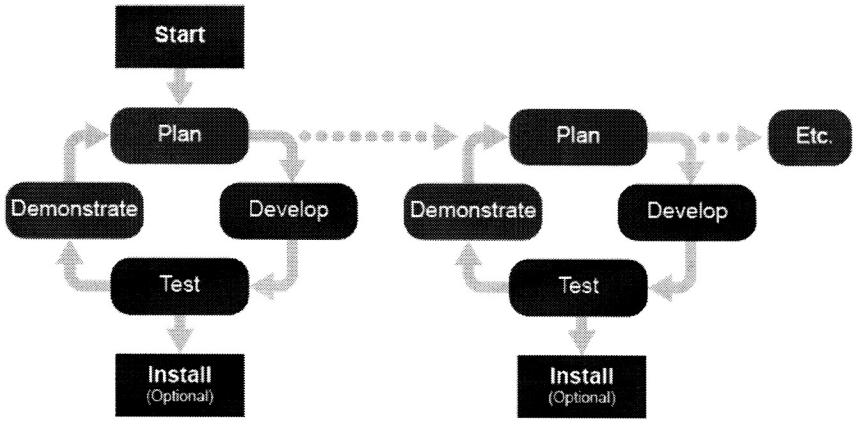
Disadvantages of the spiral model

- The process of developing prototypes, getting feedback and refining the prototypes is time-consuming so the finished product takes longer to develop
- A system is more costly to develop because of the time involved
- Not suitable for smaller projects

When to use the spiral model

- For medium to high-risk projects
- When users are unsure of their needs and what the possibilities are
- When the requirements are complex
- For large projects which may take years to develop, during which time new technologies may develop and significant changes occur

The Agile model



The Agile model

- Software is developed in rapid incremental cycles
- Each version builds on previous functionality
- Each version is thoroughly tested before release
- Good for small, time-critical projects
- Limited planning is needed to get started

Advantages of the Agile model

- Rapid, continuous delivery of useful software leads to customer satisfaction
- Customers, developers and testers constantly interact with one another
- Working software is delivered frequently, within weeks rather than months
- Software is easily adapted to changing circumstances
- Even late changes in requirements can be implemented

Disadvantages of the Agile model

- There is a lack of emphasis on necessary design and documentation
- The project can fail to deliver if the customer is not clear about the desired final outcome
- Not suitable for novice programmers – experienced programmers capable of making good decisions are required

When to use the Agile model

- When new changes need to be implemented – small incremental changes can be made frequently and for little cost
- In an expanding or developing business where users' needs are continuously changing and developing

Worksheet 1

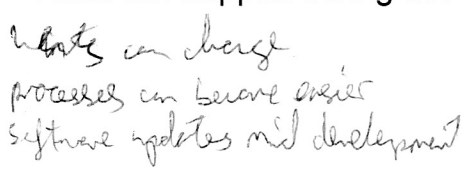
- Now try **Task 3** on the worksheet

Extreme programming

- This is a type of agile software development
- Frequent releases of the software are made in short development cycles
- It is intended to improve productivity and responsiveness to changing customer requirements

Large projects

- Some very large projects are developed over a long period of time
- What can happen during this time?



Rapid application development

- Workshops and focus groups gather requirements rather than using a formal document
- Prototyping is used to continually refine the system in response to user feedback
- Each part of the system is produced within a strict time limit – maybe not perfect, but good enough
- Software components are reused whenever possible

Consolidation

- You have looked at:
  - the waterfall lifecycle
  - the spiral model
  - agile methodologies
  - extreme programming
  - rapid application development
- Can you describe the relative merits and drawbacks of different methodologies and when they might be used?